Internal Architecture Of 8086 Microprocessor

Department of Computer Science & Engineering
BRAC University.

Course ID: CSE - 341
Course Title: Microprocessors

Lecture References:

- **Book:**
  - *Microprocessors and Interfacing: Programming and Hardware, Chapter # 2,* **Author:** Douglas V. Hall

16-Bit 8086 Intel Processor Architecture (**Important**)

Physical address = segment no x 10h + offset

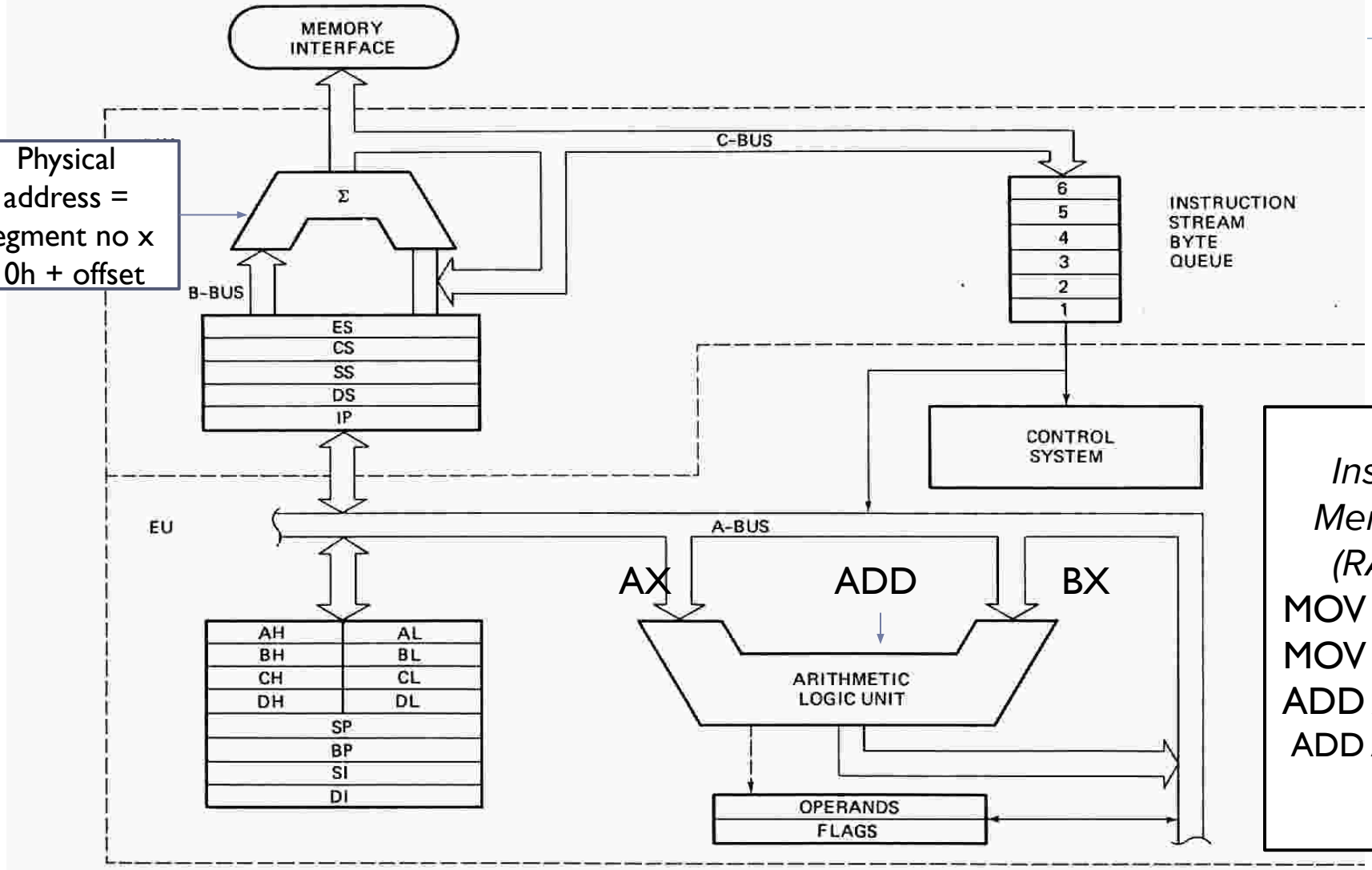


FIGURE 2-7 8086 internal block diagram. (*Intel Corp.*)

Inside Memory (RAM)

MOV AX,3H
MOV BX,4H
ADD AX,BX
ADD AX, 2H

Simply remember

Only the Segment registers (CS, DS, SS, ES) and IP are in the BIU. Every other registers are in EU.

Also the Instruction Queue is in the BIU. Other than that everything else is in EU

Organization of the 8086

- Microprocessor 8086 is internally divided into two separate functional units. These are:
  - **Bus Interface Unit (BIU)** -> Fetching
  - **Execution Unit (EU)**. -> Decoding And execution
- These two functional units can work simultaneously to increase system speed and hence the throughput.
- Throughput is a measure of number of instructions executed per unit time.
- These two units interact directly with each other through Internal Bus. (There is NO address, data and control bus within the units)

Bus Interface Unit

- **BIU:** facilitates communication between the EU & the memory or I/O circuits. It provides a full 16-bit bi-directional data bus and 20-bit address bus.
  - Responsible for transmitting addresses, data, and control signals on the buses.
  - **Registers** (CS, DS, ES, SS, and IP) hold addresses of memory locations.
  - **IP** (instruction pointer) contain the address of the next instruction to be executed by the EU.

NB: CS, DS, ES, SS are registers which hold the starting address of code, data, extra, stack segments. The segments are in the memory (RAM).

Bus Interface Unit

The bus interface unit is responsible for performing all external bus operations, as listed below.

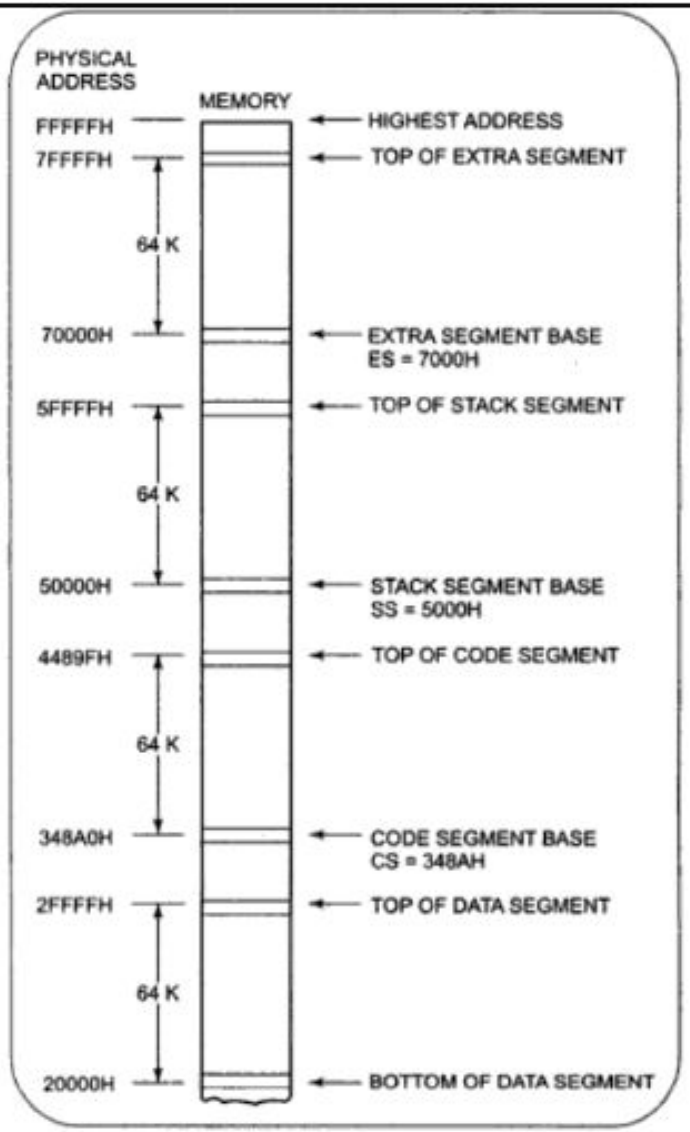
- It sends address of the memory or I/O.
- It fetches instruction from memory.
- It Reads data from port/memory.
- It Writes data into port/memory.
- It supports instruction queuing.
- It provides the address relocation facility.

Physical Address Calculation in BIU

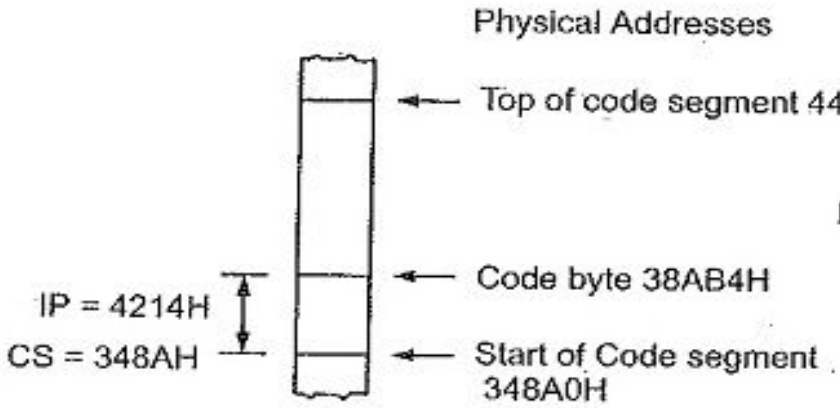
- To generate 20 bit physical address Segment Registers (CS, DS, SS, ES) and Pointer and Index Registers (IP, SP, BP, SI, DI) are needed.

Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Addresses in the stack
ES	BX, DI, SI	Address of destination data (for string instructions)

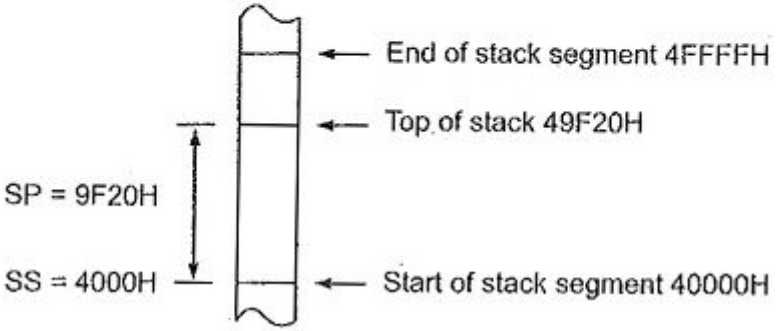
Memory Segment and Segment Registers



Physical Address Calculation Examples



SS	40000H
+ SP	9F20H
Physical address	49F20H



Physical Address Calculation Examples

The contents of the following segment registers are as given.
CS = 1111H, DS = 3333H, SS = 2526H.
IP = 1232H, SP = 1100H, offset in data segment = 0020H.
Calculate the corresponding physical addresses for the addressed byte in a) CS b) SS and c) DS.

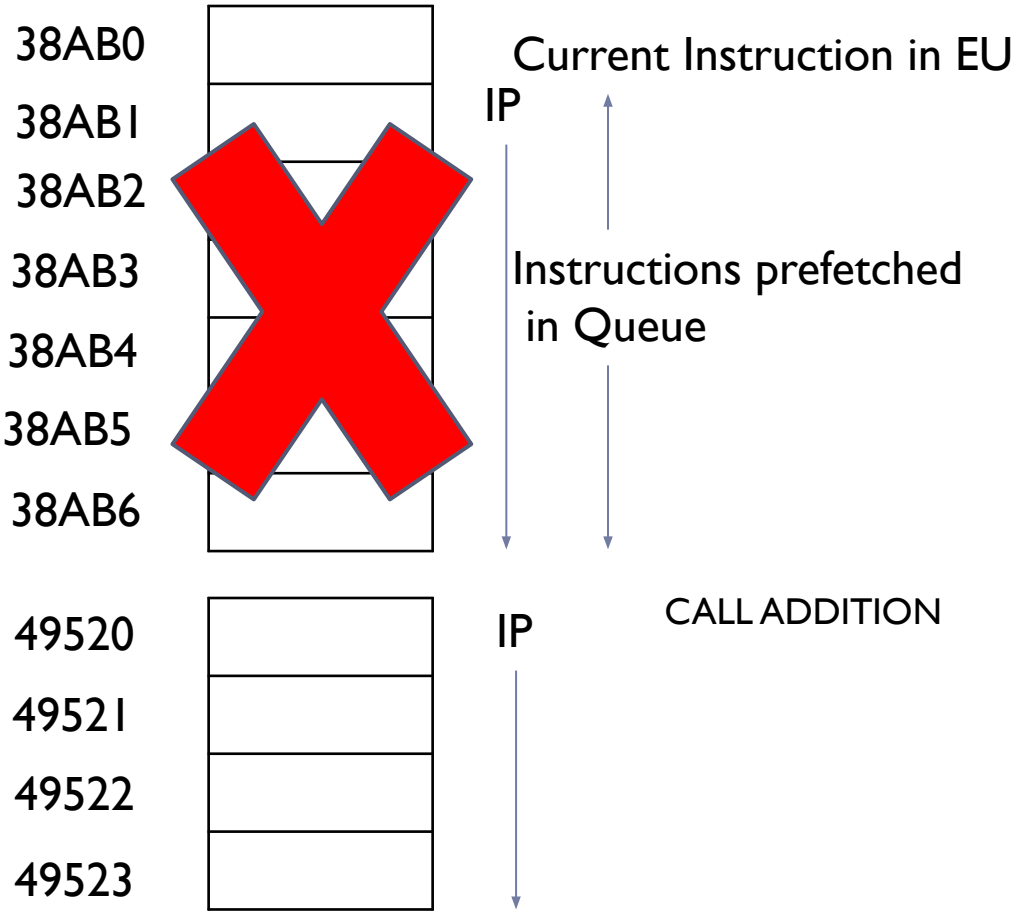
Solution

a) The base address of the code segment is 11110H. The address of the next instruction to be executed is referenced by CS and IP which is given by 11110H +1232H = 12342H.
b) The current top of the stack is referenced by SS and SP. The base address of the stack segment is 25260H. The corresponding physical address is 25260H + 1100H = 26350H.
c) The data that needs to be accessed is given by DS and the offset. The base address of the data segment is 33330H. The physical address of this data is calculated as 33330H + 0020H = 33350H.

Instruction Queue

1. To speed up program execution, the BIU fetches **6 instruction bytes** ahead of time from the memory.
2. These prefetched instruction bytes are held for the execution unit in a group of registers called **Queue**.
3. With the help of queue it is possible to fetch next instruction when current instruction is in execution.
4. **Exactly 2 Bytes** are filled up at a time.
5. The queue operates on the principle first in first out (FIFO). So that the execution unit gets the instructions for execution in the order they are fetched.
6. Exceptions: JUMP and CALL instructions

Instruction Queue



Scenario when Instruction in Queue is discarded

Lets say we have

Question?

Q) Why exactly 2 bytes are filled up at a time in Instruction Queue?

Q) What will happen if 1 byte is empty in the IQ? Will a new instruction be fetched?

Q) What will happen if an instruction is more than 2 Bytes and there is only 2 Bytes of empty space ? / Let's say an Instruction is of 4 Bytes and we have 2 Bytes of free space in IQ, will IQ partially take the instruction or discard it completely?

Pipelining to speed up execution

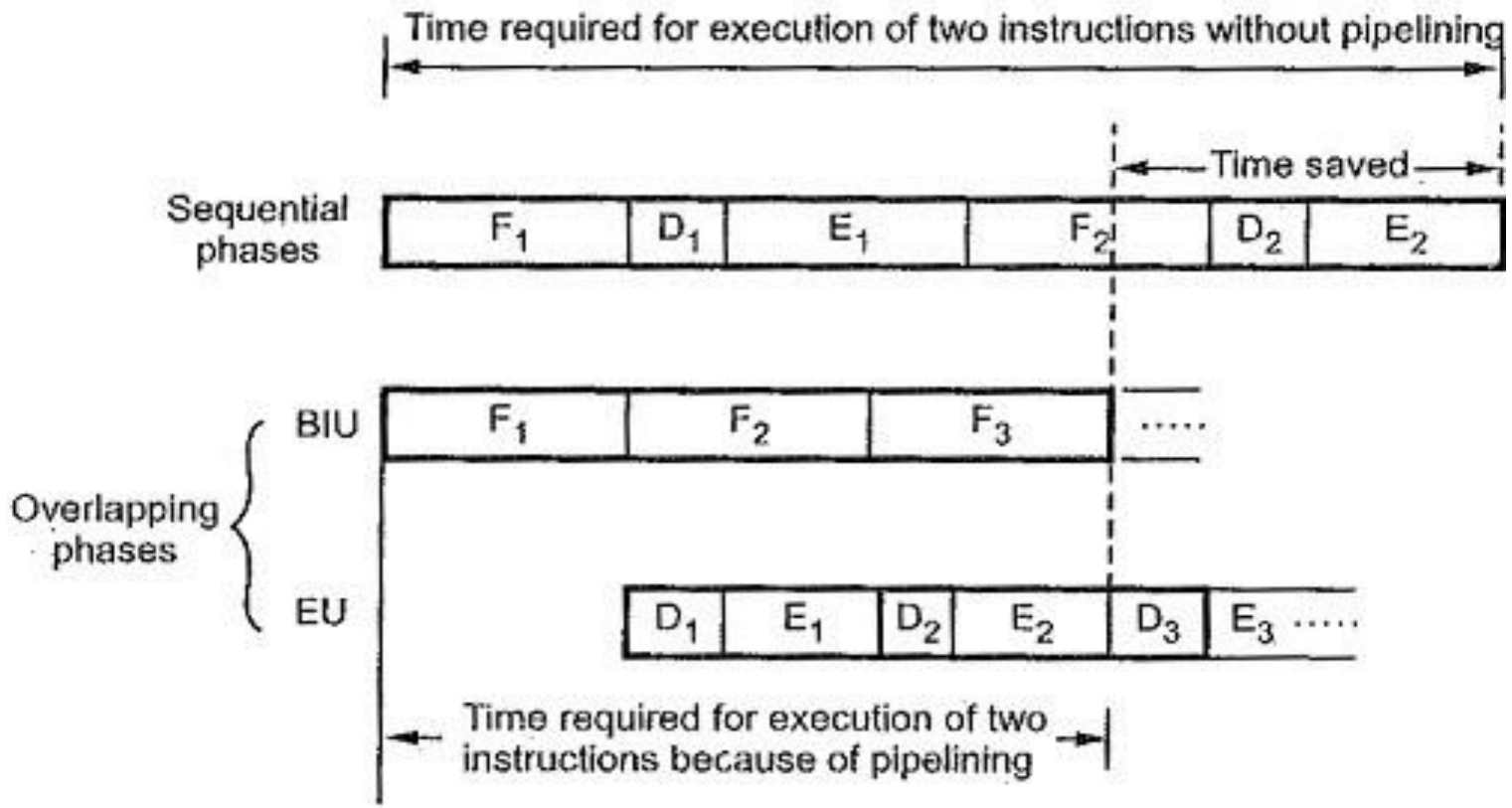


Fig: Pipelining

Question

Q1) What will happen if we remove the IQ from the BIU?

Q2) Let's say we have pipelining. But our fetching time is too high. What might happening to the execution of the process?

Q3) Now let's say our fetching time is too low. Will it change the execution time of the process.

Execution Unit (EU)

- The execution unit of Internal Architecture of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains
  1. Control Circuitry
  2. Instruction Decoder
  3. Arithmetic Logic Unit (ALU)
  4. Flag Register
  5. General Purpose Registers
  6. Pointers and Index Registers

Execution Unit (EU)

- The main components of the EU are General purpose registers, the ALU, Special purpose registers, Instruction Register and Instruction Decoder and the Flag/Status Register.
- Fetches instructions from the Queue in BIU, decodes and executes arithmetic and logic operations using the ALU.
- Sends control signals for internal data transfer operations within the microprocessor.
- Sends request signals to the BIU to access the external module.
- It operates with respect to T-states (clock cycles) and not machine cycles.

Arithmetic Logic Unit (ALU)

- This unit performs all arithmetic and logical computations.
- **Instructions** those are fetched and decoded, are executed in the ALU
- Thus, ALU has direct access to General Purpose Registers and Flags.

Temporary and Flag registers

- Every processor has some temporary registers those are not visible to the programmers.
- For these registers, our data are not lost in between different kinds of operations,
  - XCNG CX, BX
    - MOV temp, CX
    - MOV CX, BX
    - MOV BX, temp

- FLAG Registers: A flag is a flip–flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU.

Thank You !!